A Model-based Approach to the Automatic Revision of Secondary Legislation

Tingting Li
Dept. of Computer Science
University of Bath
Bath, UK
tl263@bath.ac.uk

Tina Balke
CRESS
University of Surrey
Surrey, UK
t.balke@surrey.ac.uk

Marina De Vos
Dept. of Computer Science
University of Bath
Bath, UK
mdv@cs.bath.ac.uk

Julian Padget
Dept. of Computer Science
University of Bath
Bath, UK
jap@cs.bath.ac.uk

Ken Satoh
National Institute of
Informatics
Tokyo, Japan
ksatoh@nii.ac.jp

ABSTRACT

Conflicts between laws can readily arise in situations governed by different laws, a case in point being when the context of an inferior law (or set of regulations) is altered through revision of a superior law. Being able to detect these conflicts automatically and resolve them, for example by proposing revisions to one of the modelled laws or policies, would be highly beneficial for legislators, legal departments of organizations or anybody having to incorporate legal requirements into their own procedures. In this paper we present a model based approach for detecting and finding legal conflicts through a combination of a formal model of legal specifications and a computational model based on answer set programming and inductive logic programming. Given specific scenarios (descriptions of courses of action), our model-based approach can automatically detect whether these scenarios could lead to contradictory outcomes in the different legal specifications. Using these conflicts as use cases, we apply inductive logic programming (ILP) to learn revisions to the legal component that is the source of the conflict. We illustrate our approach using a case-study where a university has to change its studentship programme after the government brings in new immigration regulations .

1. INTRODUCTION

Law is dynamic in the sense that it is constantly changing due to changes in society. It is typically changed by legislators enacting new laws or amending or repealing existing ones. One problem with changing laws is that the subjects of the law need to adapt to the new legislation. This is a problem, because depending on the changes made, their previous ways of acting might no longer be applicable or plain incorrect in the new legal setting, possibly resulting in (unintended) illegal behaviour and fines. To give an example, if employment laws change, organizations which are subject to these laws need to check whether the organization-specific employment rules are still all correct and legal under the new legislation. That is why, when laws are changed, it is essential for actors (such as the above mentioned organizations) to verify whether their existing regulations and actions are still compliant in the context of the revised legislation.

In this paper we start by modelling legal systems with the help of legal specifications. We present the formal specification and a corresponding computational model of a mechanism for automatically detecting conflicts between legal specifications, based on actors' typical course of actions. If conflicts are detected, our mechanism automatically deals with them by proposing ways to update the legal specification precedence. We use a UK immigration law case study to illustrate our approach. We stress that the approach presented is applicable to laws in general, not just the laws in the case-study. The conflict detection and resolution mechanisms are general-purpose techniques.

2. MODELLING OF LEGAL SYSTEMS

The model we use to represent legal systems has been presented in several earlier papers, amongst which the most relevant is [4]. It considers how institutional models, here called legal specifications, may be used for legal reasoning in the context of Japanese contract law. For the sake of completeness of this paper, we summarize its main features.

The approach we take is to use the model to characterize all the traces that can arise from the legally relevant actions of the parties involved in the specification. Thus a trace is a sequence of relevant actions taken by the parties involved. Each of this actions brings about a change in the legal state. Starting from the initial legal state, Δ, a trace brings about a sequence of states that is referred to as the model of the trace.

The design is of an event-driven evolution of an legal state, itself represented by a set of facts (fluents, $\mathcal{F}$). This state is acted upon through two relations: (i) $\mathcal{G}$, whose function is that of a gatekeeper, in that it determines which physical actions ($\mathcal{E}_{ex}$) are relevant to the legal specification, and conditional upon the current state, maps those to legal actions ($\mathcal{E}_{legal}$), as well as generating any violations ($\mathcal{E}_{viol}$), and (ii) $\mathcal{C}$, which updates the legal state, contingent upon the combination of the set of legal actions from the transitive closure of $\mathcal{G}$ plus any unfulfilled obligations and non-permitted events, and the current state. Also any concluded (satisfied or vio-

ICAIL '13 Jun 10-14 2013, Rome, Italy
ACM 978-1-4503-2080-1/13/06.

lated) obligations are removed. Thus, starting from an initial state Δ, a *trace* of physical actions, referred to as exogenous events, generates a sequence of states by means of a state transformer function comprising $\mathcal{G}$ and $\mathcal{C}$ as described above, resulting in a *model* of the legal specification. The legal specification formally characterized by $\mathcal{I} = \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$, namely events, fluents, the generation relation, the consequence relation and the initial state.

The formal model and its declarative specification language InstAL become a computational model by translating it to a logic program under the Answer Set semantics [8]. In answer set programming, the program is written in $AnsProlog$ as a finite set of rules of the form $r : \text{consequence} : -\text{condition}.$ or $\bot : -\text{condition}.$ A rule should be read as: "The consequence is *supported* if all components of condition are true". A rule with $\bot$ as the consequence is referred to as an *(integrity) constraint*. $\bot$ is always assigned the truth value "false". The semantics of $AnsProlog$ are defined in terms of *answer sets* – assignments of true and false to all elements of, called atoms, the program that satisfy the rules in a minimal and consistent fashion. A program has zero or more answer sets, each corresponding to a solution.

The legal specifications are mapped to a set of $AnsProlog$ rules. The rule bodies are conjunctions of literals using negation as failure for negated expressions. These rules are then combined with a set of standard rules that handle housekeeping tasks (fluent inertia, violation generation) and a set of rules for time predicates, which are generated on demand, based on the provided trace length.

Provided with a trace, the program will result in exactly one answer set containing the corresponding model of the trace. Otherwise, the program returns all traces and their corresponding models as answer sets. The program can be augmented with constraints to obtain specific traces and their models.

2.1 Modelling of Case Study

Our case study is set against the background of changes in UK immigration legislation with a particular focus on changes to student visa regulations announced on March 22nd, 2011 by the UK Home Secretary[1].

As part of these changes, the regulations concerning the permitted working hours (during studies) for international students were reduced[2].

What is vital for our case study, which for presentation purposes only concentrates on one aspect of the resulting implications, is that the reduction of international student working hours might result in conflicts with existing policies such as those for university studentships. Thus, our case study concerns the tensions between on the one hand, the regulations associated with the visa of an international student studying at a university, the limitations on how much work said visa-holder is permitted to undertake and on the other hand, the regulations associated with the studentship awarded to the student, in particular the minimum number of teaching hours that the student must deliver. In the scenario we consider here, the government reduces the number of hours that are permitted so they are less than that required by conditions of the studentship. Consequently, a conflict occurs which needs to be addressed. As the immigration law takes precedence, we solve the conflict by revising the studentship regulations to align with the government requirements, in the case of an international student. The hours for home students and EU students can remain the same or the university can decide to treat all students equally.

[1]The changes we are concerned with in this paper in particular concern §57–62 of UK Immigration Law (as of February 2011).

[2]A detailed list of these changes with the respective legal texts as well as a statement of intent can be found on the UK Home Office website under `http://www.ukba.homeoffice.gov.uk/sitecontent/documents/news/sop4.pdf`.

```
initiated(perm(work(Student,twenty)),I) :-
      occurred(intStudentApplyVisa(Student),I),
      holdsat(live(visa),I),
      holdsat(person(Student,overseas),I),
      student(Student),
      instant(I).
```

(a)

```
initiated(obl(work(Student,thirty),endOfStudy,
           withdrawStudentship),I) :-
      occurred(askStudentShip(Student),I),
      holdsat(availability,I),
      student(Student),
      instant(I).

initiated(perm(work(Student,thirty)),I) :-
      occurred(askStudentShip(Student),I),
      holdsat(availability,I),
      student(Student),
      instant(I).
```

(b)

Figure 1: $AnsProlog$ code fragments from the visa regulations (a) and the university regulations (b)

Table 1 provides the $\mathcal{G}$ and $\mathcal{C}$ functions in the formal model for our case study.

In the $AnsProlog$ translation we use corresponding atoms. The relation between a student and his/her nationality is represented by a domain fluent `person(Student,Nationality)`. For a concrete scenario, `person(tingting,overseas)` for example, would be added to the initial state. The main obligation fluent is: after accepting an offer `acceptOffer(Student)`, an overseas student is obliged to apply for a visa `applyStudentVisa(Student)`. Failure to do so results in a violation event `illegalImmigrant(Student)`. Once this student has applied for the visa, he or she will be given a Tier4 visa and permission to work for twenty hours per week (`perm(work(Student,twenty))` resp.). Home and EU students automatically have the permission to work for twenty, thirty hours or full-time. This information is directly encoded as part of the initial state. Figure 1 provides an $AnsProlog$ code fragment of each legal specification that includes the rules that are causing the conflict. Given the supremacy of the immigration law, it is the rules from the university specification that need to be revised.

3. LEGAL CONFLICT DETECTION

Having described how legal specifications can be used to model laws and legal systems, this section turns the focus to detecting conflicts between two legal specifications. The definition of conflicts and the mechanism for conflict detection have initially been presented in our previous work [9]. Based on that, in this paper we extend the definition to distinguish between weak and strong conflicts depending on the nature of the conflicts.

3.1 Preliminary Definitions

In order to detect the disparities among a set of individual legal systems or laws, we model them as legal specifications using the approach of Section 2. Together they form a so-called *composite legal specification*, denoted as $C_{\mathcal{L}}$.

We assume that there is a shared legal ontology within $C_{\mathcal{L}}$. If the original specifications did not use the same legal ontology, they would have been semantically aligned before they were put together as a composite.

In other words, the same concept (e.g an action or a piece of domain information) is denoted by the same $AnsProlog$ represen-

Visa Regulations	University Policy
$\mathcal{G}(\mathcal{X},\mathcal{E})$: $\langle \emptyset, \texttt{applyStudentVisa(Student)} \rangle \rightarrow$ $\{\texttt{intStudentApplyVisa(Student)}\}$ $\mathcal{C}^{\uparrow}(\mathcal{X},\mathcal{E})$: $\langle \{\texttt{person(Student, overseas)}\}, \texttt{acceptOffer(Student)} \rangle \rightarrow$ $\{\texttt{obl(applyStudentVisa(Student), arrival(Student), illegalImmigrant(Student))}, \texttt{pow(intStudentApplyVisa(Student))}, \texttt{perm(intStudentApplyVisa(Student))}\}$ $\langle \{\texttt{person(Student, overseas)}\}, \texttt{intStudentApplyVisa(Student)} \rangle \rightarrow$ $\{\texttt{studentVisa(Student, tier4)}, \texttt{perm(work(Student, twenty))}\}$ $\mathcal{C}^{\downarrow}(\mathcal{X},\mathcal{E}) = \emptyset$	$\mathcal{G}(\mathcal{X},\mathcal{E})$: $\langle \emptyset, \texttt{applyToStudy(Student, Nationality)} \rangle \rightarrow$ $\{\texttt{intApplication(Student)}\}$ $\mathcal{C}^{\uparrow}(\mathcal{X},\mathcal{E})$: $\langle \emptyset, \texttt{intApplication(Student)} \rangle \rightarrow$ $\{\texttt{obl(sendOfferLetter(Student), startofTerm, invalidOffer)}\}$ $\langle \{\texttt{person(Student, Nationality)}\}, \texttt{intApplication(Student)} \rangle \rightarrow$ $\{\texttt{offer(Student, Nationality)}\}$ $\langle \emptyset, \texttt{sendOfferLetter(Student)} \rangle \rightarrow$ $\{\texttt{sendOfferLetter(Student)}\}$ $\langle \{\texttt{offer(Student, Nationality)}\}, \texttt{acceptOffer(Student)} \rangle \rightarrow$ $\{\texttt{offerConfirmed(Student, Nationality)}, \texttt{perm(askStudentShip(Student))}\}$ $\langle \{\texttt{availability}, \texttt{askStudentShip(Student)}\}, \texttt{intComp(Promissor, Promisee, ContractID, AgreedFine)} \rangle \rightarrow$ $\{\texttt{obl(work(Student, thirty), endOfStudy, withdrawStudentship)}, \texttt{perm(work(Student, thirty))}\}$ $\mathcal{C}^{\downarrow}(\mathcal{X},\mathcal{E}) = \emptyset$

Table 1: The visa and university laws and procedures with repect to working hours of students

tation.

3.1.1 Composite Traces

Having formed a *composite legal specification* $C_{\mathcal{L}}$ from a set of legal specifications, we now can continue with the conflict analysis within $C_{\mathcal{L}}$. For this purpose, the composite specification will be analysed by event traces specified by the user containing the actions of the stakeholders in order to ascertain whether the system is conflict-free or not.

These traces are referred to as *composite traces* (*CTR*) and are formed by exogenous events from any individual specification in $C_{\mathcal{L}}$. This implies that each event can be recognized by one or more individual specifications, but not necessarily by all of them.

DEFINITION 1. *Given a composite legal specification $C_{\mathcal{L}}$, formed from individual specifications $\{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, a* composite trace*, denoted as* CTR*, is a event sequence $\langle e_1, \ldots e_m \rangle$ such that $\forall e_i, 1 \leq i \leq m : \exists 1 \leq j \leq n : e_i \in \mathcal{E}^j_{ex}$, denoted as* CTR.

From the definition, *composite traces* can describe all possible scenarios in the context of a $C_{\mathcal{L}}$, which enables: (i) a comprehensive and general analysis of all potential conflicts between specifications in $C_{\mathcal{L}}$ by generating all *composite traces* of $C_{\mathcal{L}}$, and (ii) a user-lead conflict analysis, by selecting those composite traces that are most relevant with respect to the stakeholders. Use cases can also cover these case specified by the designer. Full coverage can be achieved, although possible computationally expensive, by not providing the system with a use case. The system will then compute all conflict traces of a certain specified length.

3.1.2 Synchronization with Null Events

Given a composite trace *CTR* of $C_{\mathcal{L}}$, the state transition of each legal specification in $C_{\mathcal{L}}$ is driven by the events appearing in the *CTR*. Since not every exogenous event is necessarily recognised by each individual specification, they can become desynchronised making comparison difficult. To solve this problem technically, we add a special event, called *null event* (e^i_{null}) to progress the state transition of individual specification $\mathcal{L}_i$ at time t if the occurred event in the CTR at time t is unknown to $\mathcal{L}_i$. It should be noted that *null events* only increase the legal state counter, but do not cause any change to the state itself.

DEFINITION 2. *Given a composite legal specification $C_{\mathcal{L}}$ with $C_{\mathcal{L}} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, a composite trace $CTR = \langle e_1, \ldots, e_t \rangle$ for $C_{\mathcal{L}}$. The event sequence $\langle a_1, \ldots, a_t \rangle$ is a* synchronized trace*, for $\mathcal{L}_i \in C_{\mathcal{L}}$ if $a_k = e_k$ when $e_k \in \mathcal{E}^i_{ex}$ or $a_k = e^i_{null}$ otherwise.*

With synchronised traces, we can generate a *composite model* for $C_{\mathcal{L}}$, as a set of models corresponding to the synchronised traces of the individual specifications L_i, denoted as $\mathcal{M}_i$.

3.2 Legal Conflicts and Conflict Traces

Legal conflicts are detected by comparing the states of each individual specification at a given time instant. The composite traces which result in legal conflicts are referred to as *conflict traces*.

We distinguish between two types of legal conflicts: *weak* and *strong conflicts*. A weak conflict between two laws is defined as the existence of a common fluent between any pair of individual specifications that is true in one but false in the other. For instance, a weak conflict can be identified when the permission fluent concerning working in the UK holds true in one legal specification but is false in another one at the same time.

DEFINITION 3. *Given a composite trace CTR for a composite legal specification $C_{\mathcal{L}}$, the CTR is a* weak conflict trace *iff:*

- $\exists \mathcal{L}_i, \mathcal{L}_j \in \mathcal{L}$ *with synchronised models* $\mathcal{M}_i = \langle S^i_0, \ldots S^i_t \rangle$ *and* $\mathcal{M}_j = \langle S^j_0, \ldots S^j_t \rangle$ *such that*
- $\exists f \in (\mathcal{F}^i \cap \mathcal{F}^j)$ *such that*
- $\exists k, 0 \leq k \leq t$ *such that*
- $f \in S^i_k$ *and* $\neg f \in S^j_k$

We have a *strong conflict* when there is simultaneously an event that is not permitted in one specification while in another there is an obligation for the event.

DEFINITION 4. *Given a composite trace CTR for a composite legal specification $C_{\mathcal{L}}$, CTR is a* strong conflict trace *iff:*

- $\exists \mathcal{L}_i, \mathcal{L}_j \in \mathcal{L}$ *with synchronised models* $\mathcal{M}_i = \langle S^i_0, \ldots S^i_t \rangle$ *and* $\mathcal{M}_j = \langle S^j_0, \ldots S^j_t \rangle$ *such that*
- $\exists e \in \mathcal{E}^i \cup \mathcal{E}^j$
- $\exists p \in \mathcal{P}^i$, $p = perm(e)$ *such that*
- $\exists o \in \mathcal{O}^j$, $o = obl(e, d, v)$ *such that*
- $\exists k, 0 \leq k \leq t$ *such that*
- $o \in S^j_k$ *and* $\neg p \in S^i_k$

A composite legal specification is *conflict-free* if it does not admit either weak or strong conflict traces.

3.3 Automatic Conflict Detection

Having presented the theoretical side of conflicts, we now turn to the computational side. With the help of legal specification, conflicts can be detected automatically by comparing the inconsistent states among their models $\mathcal{M}$ for any synchronised event traces at any given time. The brute force method of combining the $Ans\text{-}Prolog$ programs of legal specifications together with the rules `conflict` : − `holdsat(F,T)`, **not** `holdsat(F,T)`. and : −**not** `conflict` cannot give any result, as the body is contradictory.

Since the problem is caused by the *syntactic* identity of F, it can be resolved by a systematic renaming of literals, as long as we remember what has been renamed as what. Thus, we add rename(F, FX, X) for each event and fluent in the individual legal specification of $C_{\mathcal{L}}$, where rename/3 indicates that an event or a fluent F is renamed FX in the legal specification X. We now present the main parts of detection program $P_{C_{\mathcal{L}}}$:

Weak Conflicts:

```
conflict(FX,FY,I) :- holdsat(FX,I), not holdsat(FY,I),
                     rename(F,FX,X), rename(F,FY,Y),
                     ifluent(FX), ifluent(EY),
                     instant(I), inst(X;Y).
```

Strong Conflicts:

```
conflict(FX,FY,I) :- holdsat(obl(FX,D,V),I),
                     not holdsat(perm(FY),I),
                     rename(F,FX,X), rename(F,FY,Y),
                     oblifluent(obl(FX,D,V)), inst(X;Y),
                     instant(I), ifluent(perm(FY)).
```

Conflict Selection:

```
conflict : -conflict(FX,FY,I).
         : -not conflict.
```

Combining this with a program P_t that specifies the conditions for what makes a trace of a certain length, we obtain all conflict traces and models as answer sets. If we have a trace CTR, we can create the program P_{CTR} containing the facts that stipulate that these events have been observed and include the necessary time atoms. Combining this with $P_{C_{\mathcal{L}}}$(the $AnsProlog$ program of a $C_{\mathcal{L}}$), we obtain a single answer set if the trace is a conflict trace.

4. LEGAL CONFLICT RESOLUTION

4.1 Conflict Resolution via Inductive Learning

Having determined (weak and strong) conflict traces in a composite legal specification $C_{\mathcal{L}}$ and obtained the literals conflict/3 capturing the information about the detected conflicts, we can now move on to resolve these conflicts. We propose to do so by revising inferior laws. We take the basic ILP mechanism set out in [3] and extend this technique to composite legal specification – instead of just one – and to synthesize the use cases for the learning specification automatically – instead of manually – following conflict detection. In consequence, a partial solution is the revision of the inferior legal specification (L_z) such that the conflict no longer occurs when the composite trace is used as input to the revised $C_{\mathcal{L}}$. As in the individual case, we are interested in the minimal revision, which is expressed in terms of distance. A complete solution requires a minimal revision for each of the conflicts.

DEFINITION 5. *The* legal conflict resolution task *can be denoted as* $\langle C_{\mathcal{L}}, \succ_{C_{\mathcal{L}}}, CTR, M\rangle$ *where* $C_{\mathcal{L}}$ *is a composite legal specification with a set of individual legal specifications among which the precedence order is defined as* $\succ_{C_{\mathcal{L}}}$. CTR *is a conflict trace which admits a set of conflicts* $\mathbb{C}_{tr}$. M *is a set of mode declarations such that* $\forall L_i \in C_{\mathcal{L}} \cdot L_i \subseteq 2^M$. *A composite legal specification* $C'_{\mathcal{L}}$ *is* partial solution *to the task if* $\exists c = \texttt{conflict(FX;FY;I)} \in \mathbb{C}_{tr}$ *with* $L_z = L_x$ *iff* $L_y \succ L_x$ *or* $L_z = L_y$ *otherwise.*
(i) $\forall L_i \in C_{\mathcal{L}}, L_i \neq L_z \cdot L_i \in C'_{\mathcal{L}}$ *(ii)* $C'_{\mathcal{L}} \cup CTR \models \neg c$ *(iii) distance between Z and Z' is minimal. A composite legal specification* $C'_{\mathcal{L}}$ *is a* complete solution *if it is a partial solution of each conflict* $c \in \mathbb{C}_{tr}$

From the definition, given a conflict c between the legal specification L_x and L_y, we label the one with lower precedence as L_z, then we revise L_z to be L'_z. The revised composite legal specification $C'_{\mathcal{L}}$ is formed from L'_z and the other (unchanged) legal specifications, and is guaranteed not to lead to the conflict c. Moreover, to minimise the side effects of the revision, we select the minimal revision as measured by the number of changes. In order to obtain the solutions to the conflict resolution task, we employ the inductive learning method introduced by [3].

Given a composite legal specification and a conflict trace CTR, we can rephrase the conflict resolution task as a theory revision task $\langle P, B, T, M\rangle$ such that the solution of the revision results in a partial solution $C'_{\mathcal{L}}$ by means of the following steps:

1. $P = \{\neg c \cup CTR\}$: The revised composite should no longer exhibit c when the trace is used. So the conflict negation and the trace become the properties for the theory revision.
2. $Rev = \{L_x \mid \texttt{conflict(FX,FY,T)} \vee \texttt{conflict(FY,FX,T)}\ s.t. \texttt{L}_\texttt{y} \succ \texttt{L}_\texttt{x}\}$: This set contains all the legal specification that need to be revised.
3. $B = \{L_i \mid L_i \in C_{\mathcal{L}}, L_i \notin Rev\}$: The base theory consists of those legal specification in the composition that are unchanged.
4. $T = \{L_x \mid L_x \in Rev\}$: Those legal specifications marked for revision will form the theory that needs to be revised.
5. $M = \{M_x \mid L_x \in Rev\}$: the mode declarations for the revision are based on the legal specifications that are marked for revision. Since conflicts are related to the mode declarations should focus on those rules in the legal specifications that have an effect on the state, which means the rule dealing with $\mathcal{G}$ and $\mathcal{C}$. Therefore, the patterns of head and body mode declarations should follow their structure: head mode declarations are associated with legal actions and fluents, while the body mode declarations are defined for all events and fluents. We use declarations to indicate which parts need to be added or deleted.

Using our own implementation *LC-RES* of Corapi et al's system *ASPAL*, we can compute our revised composite legal specification. Here we only provide an brief overview of the specification. A more detailed description of theory can be found in [2] and [3]. With the help of mode declaration, *revision tuples* for those legal specifications that need to be revised can be generated. These denote (i) all acceptable revisions to the existing rules, and (ii) all acceptable new patterns of rules. The revision tuples are represented as a literal rev/4 that indicates which rule needs which kind of revision and its associated cost.

The result of the revision specification will be a set of answer sets containing some of these revision tuples. Each answer set provides an alternative revision of the legal specification. Since we are only interested in the revision with the minimum cost, we adopt the *aggregate* statement, provided by the answer set solver CLINGO [7], which has a lower and upper bound by which the weighted literals can be constrained. Therefore, we append the ASP rule $:- \textbf{not}\ [\texttt{rev}(\_,\_,\_,\texttt{Cost}) = \texttt{Cost}]\texttt{Max}$ to each revisable theory. We apply an incremental strategy for the variable Max, for example, if no solution can be found when $\texttt{Max} = 1$, then we continue with $\texttt{Max} = 2$ and so on until a solution is found. While this is currently done manually, we could with little changes move to the incremental solver ICLINGO [6].

4.2 Conflict Resolution in Case Study

We now resolve the conflicts detected in the case study by means of *LC-RES*. The input is a composite legal specification and a set of *use cases*. For simplicity, we associate a distance cost with addition and deletion to the *Cost* argument in revision tuples, but this can be adjusted to give different outcomes, via the optimization mechanism discussed above. By having the precedence of *Visa Regulation* higher than *University Policy*, the optimal solution obtained is as below, which implies a revision to *Univeristy Policy* in order to be

compliant with *Visa Regulation*:

$$\texttt{rev}(\texttt{university}, 3, \texttt{r}(\texttt{hINIT3}, \texttt{bHO3}, \texttt{l}(0)), 1)$$
$$\texttt{rev}(\texttt{university}, 2, \texttt{r}(\texttt{hINIT2}, \texttt{bHO3}, \texttt{l}(0)), 1)$$

The result suggests a revision to the rule 3 and rule 2 by adding the domain fluent $\texttt{person}(\texttt{Student}, \texttt{overseas})$ with mode declaration id bHO3, then rule 3 and 2 become:

```
initiated(perm(workUNI(Student, thirty)), I) :-
          occurred(askStudentShipUNI(Student), I),
          holdsat(availabilityUNI, I),
          not holdsat(person(Student, overseas), I).

initiated(obl(workUNI(Student, thirty), endOfStudyUNI,
                    withdrawStudentshipUNI), I) :-
          occurred(askStudentShipUNI(Student), I),
          holdsat(availabilityUNI, I),
          not holdsat(person(Student, overseas), I).
```

Therefore, the university regulation should revise the rule as: for all non-overseas students, the permission and the obligation to work thirty hours per week are initiated.

5. RELATED WORK

Detecting and resolving legal conflicts are not new issues and have been investigated for several decades (a comprehensive discussion is provided in [10]). [11] addresses both detection and resolution of normative conflicts by means of static comparison using *first-order unification* to obtain overlapping substitution values to the variables appearing in a pair of norms/rules with contradictory normative positions, e.g. permission and prohibition, obligation and prohibition. In contrast to their work, the conflicts we address, shift attention to a broader level of legal specifications, which implies that they may occur implicitly and cannot be observed easily by the static comparison of legal rules. Therefore, we detect the legal conflicts through interacting with a sequence of events (i.e. composite traces) and continuous comparison of the changing legal states and consequences, which makes the approach more comprehensive and of practical use. [5] and [10] present a formal definition and analysis of conflicts, but no computational mechanisms are provided. Moreover, their work also assume that all conflicts are known *a priori*. However, it is very likely that individuals do not have sufficient legal knowledge to be aware of potential conflicts arising from their cases. So our approach works both for specific traces but can also detect conflicts without them.

Regarding the mechanism for conflict resolution, Vasconcelos et al. annotate each rule with an *undesirable set* containing values that cause conflicts, then the rule with lower precedence has to avoid being assigned these values. An alternative method is provided by [10], where either a belief change function or a non-monotonic reasoning approach is applied with regards to classic ordering strategies over laws. García-Camino et. al [5] achieve conflict resolution by removing laws with lower precedence. Contrasted to the existing work above, there are two novel contributions in our resolution mechanism: (i) our strategy is to apply a minimum revision to the legal specification with lower priority, making it consistent with the others, rather than ignoring or deleting them, and (ii) the LC-RES tool is fully implemented under $AnsProlog$ and supported by ASP solver CLINGO, providing an automated revision mechanism.

6. CONCLUSIONS AND FUTURE PLAN

Conflicts of law might occur between legal systems from different jurisdictions (e.g. different countries), or also might be caused unintentionally between new legislation and existing laws (e.g. as in the scenario described in the case study). It is of great importance for individuals on the one hand, to be able to foresee potential conflicts as consequence of a sequence of events, and on the other hand, for legislators and rule makers to know how to prevent the occurrences of conflicts when revising current laws. In this paper, we first presented a formal and computational approach supporting legal modelling and reasoning. Based on that, we then introduced an approach for automatically detecting conflicts between different legal systems, which is able to not only detect conflicts at a general level of a composite legal specification, but also to identify conflicts that are caused by specific cases (i.e. traces of exogenous events). Furthermore, in order to resolve the conflicts, we implement an automatic mechanism based on *inductive learning* to produce revision suggestions with minimum cost to the current legal systems.

For the future work, we outline several issues and extensions to the current approach: (i) establish a legal ontological framework to relax the assumption of semantic alignment within a composite legal specification. (ii) extend the method to handle multiple traces and even all possible traces by giving each trace separate unrelated state instances. (iii) investigate further the complete solution to resolve all conflicts.

7. REFERENCES

[1] O. Cliffe, M. De Vos, and J. A. Padget. Answer set programming for representing and reasoning about virtual institutions. In K. Inoue, K. Satoh, and F. Toni, editors, *Computational Logic in Multi-Agent Systems*, volume 4371 of *LNCS*, pages 60–79. Springer, 2007.

[2] D. Corapi. *Nonmonotonic Inductive Logic Programming as Abductive Search*. PhD thesis, Imperial College London, 2011.

[3] D. Corapi, A. Russo, M. D. Vos, J. A. Padget, and K. Satoh. Normative design using inductive learning. *TPLP*, 11(4-5):783–799, 2011.

[4] M. De Vos, J. Padget, and K. Satoh. Legal modelling and reasoning using institutions. In T. Onoda, D. Bekki, and E. McCready, editors, *New Frontiers in Artificial Intelligence (JSAI-isAI 2010 Workshop)*, pages 129–140. Springer, 2011.

[5] A. García-Camino, P. Noriega, and J.-A. Rodríguez-Aguilar. An algorithm for conflict resolution in regulated compound activities. In *Engineering Societies in the Agents World VII*, volume 4457, pages 193–208. Springer, 2007.

[6] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107–124, 2011.

[7] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-Driven Answer Set Solving. In *Proceeding of IJCAI07*, pages 386–392, 2007.

[8] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):365–386, 1991.

[9] T. Li, T. Balke, M. De Vos, K. Satoh, and J. Padget. Conflict detection in composite institutions. In *International Workshop on Agent-based Modeling for Policy Engineering (AMPLE 2012)*, page 66, 2012.

[10] G. Sartor. Normative conflicts in legal reasoning. *Artificial Intelligence and Law*, 1:209–235, 1992.

[11] W. Vasconcelos, M. Kollingbaum, and T. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, 2009.